Getting Started with Arrays and DO LOOPS

Created 01/2004,
Updated 03/2008,
Afton–Royal Training & Consulting

What to Expect

We have 30 minutes to learn how to use DO LOOPS and arrays. We will assume you already know the basic data step techniques to prepare your data, and that you have some goal of what you will do with it later.

- First, we'll go over the different uses of DO.
- Second, we'll look at how to create an array.
- Third, we'll learn how to use one dimensional arrays, and see a list of resources for further reading.

Section 1: DO, DO WHILE, DO UNTIL

The DO Statement

- A DO statement designates a group of other statements to be run by SAS.
- An END statement shows where the group ends – you will get an error and no results if you leave it out.
- DO statements can be nested – be careful to match up the END statements correctly.
- Normally, the DO is in the middle of the statement, after a condition, but with arrays this is not as common.

Example: DO Statement

```
data one.good one.bad;
   set one.all;
      if risk GE 8 then do;
         RiskCategory='High';
         output one.bad;
      end;
      else if 1 GE risk LE 4 then do;
         RiskCategory='Low';
         output one.good;
      end;
      else if 4<risk<8 then
         RiskCategory=‘Med’;
run;
```

The Iterative DO Statement

- You can iterate, or loop, through a block of statements (DO through END) for a specified number of iterations, by adding an index variable specification.
- Because it doesn't matter what the index variable is called as long as you don't reuse the name of an existing variable by accident, most people use "i" – this variable is saved in your data unless you drop it.
- The simplest and most common specification is "i=1 to N" with N as the number of iterations you want.

Examples: Iterative DO

Most common: start to stop (by 1)

```
do i=1 to 5;
do i=6 to 10;
do i=1 to limit; *variable*;
do i=1 to count-1; *expression*;
```

Unusual, but fine

```
do i='one','two',‘four'; *char list*;
do count=1,3,5,7,9;  *num list*;
do odds=1 to 9 by 2;  *BY increment*;
do i= 10 to -4 by -.5;  *backwards*;
```

Infinite Looping

- Sometimes you will accidentally cause infinite looping, but it can be avoided.
- Be careful when changing the value of the index variable inside the loop. Think through the effects before running.
- Don’t try to change the value of START during the looping, because it is evaluated when the loop begins.
- To get out of a DO loop, use LEAVE.

DO UNTIL and DO WHILE Statements

- After understanding the syntax of the simple DO and iterative DO statements, the syntax of the DO UNTIL and DO WHILE statements is simple to learn.
- The condition to be evaluated goes in parentheses after DO UNTIL or DO WHILE.
- The end is used as before.
- You must change the value of the index!
- The condition is evaluated
    1. after the loop with UNTIL,
    2. before the loop with WHILE.

Example: WHILE vs UNTIL

```
*until evaluates after loop*;
   do until (x>=3);
      put x= 'until ' _n_=;
      x+1; *evaluates now*;
   end;
*while evaluates before loop*;
  do while (p<=3); *evaluates now*;
      put p= 'while ' _n_=;
      p+1;
   end;
```

Summary: DO

- The DO statement designates a block of statements to execute (ending with “end;”).
- The iterative DO statement executes the block multiple times, based on an index.
- DO WHILE evaluates a condition in parentheses, and then executes the block if the condition is true, repeatedly.
- DO UNTIL executes the block, then evaluates the condition and repeats the block if the condition is true (you always get at least 1).

Section 2: Array Syntax

Why Use an Array?

- When you need to do the same thing to a whole list of variables.
- When you find yourself cutting and pasting code many times, then changing the variable names.
- When you want to check the values of a group of variables that should be similar.

What is an Array?

- A temporary grouping of variables, in a specified order
- Only exists in the data step
- Does not carry over to the next data step
- The array itself is not a variable.
- The array must be defined before you can use it.

Defining an Array

- Use an ARRAY statement to define an array. It begins with the word “array”.
- Next, name the array (can’t be a variable name, and shouldn’t be a function name).
- Third, put the number of elements in parentheses of some sort.

```
ARRAY name{sub} <element1 element2, …>;
```

The Elements of an Array

- If you want the elements of the array (variables) to be named after the array itself, you don't need to list them.

```
ARRAY Bob(3); *elements are Bob1, Bob2, Bob3*;
```

- If you want the elements to have any other names, you must list the variables **in order**.

```
ARRAY Jim(3) Jim1 Jim6 JimBob;
```

- If the elements will be character variables, and they haven't yet been defined, you must add a $ after the array name and subscript.

```
ARRAY Fred(2) $ FirstName LastName;
```

Initial Values and Temporary Arrays

- If you are creating array elements and want to initialize them with values, put the value list in parentheses after the element list.

```
ARRAY Mary(2) $ MaryJane MaryBeth ('yes' 'no');
```

- When you are creating array elements that you won't need in the future, it is best to use a temporary array, which does not create variables.
  - Add _TEMPORARY_ before the values.
  - Do not list element names.
  - The values are automatically retained.

```
ARRAY Mary(2) $ _temporary_ ('yes' 'no');
```

Using an Array

- After the ARRAY statement which creates the array, you can reference the array and it's elements easily.
- To reference the entire array, use the array name.
- To refer to a specific element of an array, use the index value(s) with the name – `Mary{1}`.
- To refer to each of the elements of the array in turn (in an iterative DO loop), use an index variable with the name – `Mary{i}`.

Example: One Array

```
*Define the array first*;
array stats(3) age weight height;

*Use the array in an iterative DO loop*;
do i=1 to 3;
   if stats(i)=0 then do;
      stats(i)=.;
      count+1;
   end;
end;
```

Example: Two Arrays

```
*Define the arrays first*;
array stats(3) age weight height;
array good(3) GoodAge GoodWt GoodHt;

*Use the arrays in an iterative DO loop*;
do i=1 to 3;
   if 0<stats(i)<900 then
      good(i)=1;
end;
```

Resources

- The best resource to ask for help with SAS is the person who was doing your work before you – “borrow” code when you can.
- Helpful reading

  (from http://v9doc.sas.com/sasdoc/):
  - **SAS Language Reference: Concepts**
    - **Chapter 31, Array Processing**
  - **SAS Language Reference: Dictionary** – look for do statement, and array statement